• Input: a binary tree where every node has a weight

• For each node, compute the sum of the weights in the subtree rooted at the node.

• The following randomized parallel algorithm runs in $O(\log n)$ steps with very high probability.

1 Algorithm

1. Every node randomly picks one of its neighbors in the following way:
   - Leaves always pick their parents.
   - Nodes with 1 child pick its parent or its child with probability 1/2 each.
     - If the root has 1 child, it always picks its child.
   - Nodes with 2 children pick either of its children with probability 1/2 each.

2. If two nodes picked each other, the child node is merged into the parent node.
   - The child node disappears, and the parent node increments its weight by the weight of the child node.
   - The parent node now represents the union of the two nodes and has the weight that is the sum of the original weights of the nodes that it represents.
   - If the child node had a child, the parent node “inherits” the child.
     - A node with two children cannot be contracted away.
     - The resulting tree is still a binary tree.

3. Repeat steps 1 and 2 recursively until only the root is left.
   - At this point, the root’s weight is the sum of all original weights in the tree.

4. Expand the tree by reversing the contractions.
   - If a node had a child immediately before it was contracted away, increment the weight of the node by the weight of the child.
     - We prove by induction that the node now has the correct sum of the original weights in the subtree rooted at the node.
2 Runtime Analysis

- At any iteration of the algorithm, let \( v_0, v_1, \) and \( v_2 \) be the number of nodes with 0, 1, and 2 children, respectively.

- We can show that \( v_0 = v_2 + 1 \).

- At any iteration, the expected number of nodes that disappear is at least \( v_0/2 + v_1/4 > (v_0 + v_1 + v_2)/4 \), which is 1/4 of the nodes.

- We call an iteration successful if at least 1/8 of the nodes disappear. An iteration is successful with probability at least 1/3.

- \( O(\log n) \) successful steps are sufficient to contract the tree, i.e. there exists a constant \( c \) such that \( c \log n \) successful steps are sufficient for all \( n \).

- For any constant \( c' \), the probability that the algorithm does not terminate within \( c' \log n \) iterations is at most the probability that fewer than \( c \log n \) of the \( c' \log n \) iterations are successful.

- We use the Chernoff bound to bound the probability that fewer than \( c \log n \) iterations are successful.

  - Chernoff bound: Let \( X \) be the sum of independent indicator random variables. If \( \mu \leq E[X] \) and \( 0 < \delta < 1 \), \( \Pr(X < (1 - \delta)\mu) < e^{-\mu\delta^2/2} \).
  
  - Let \( X \) be the sum of \( c' \log n \) indicator variables that indicate whether an iteration was successful. Then, \( E[X] \geq (c' \log n)/3 \). Let \( \mu = (c' \log n)/3 \).
  
  - Set \( \delta \) such that \( (1 - \delta)\mu = c \log n \). \( \delta \) is an increasing function of \( c' \).
  
  - \( e^{-\mu\delta^2/2} \) decreases exponentially with respect to \( c' \).